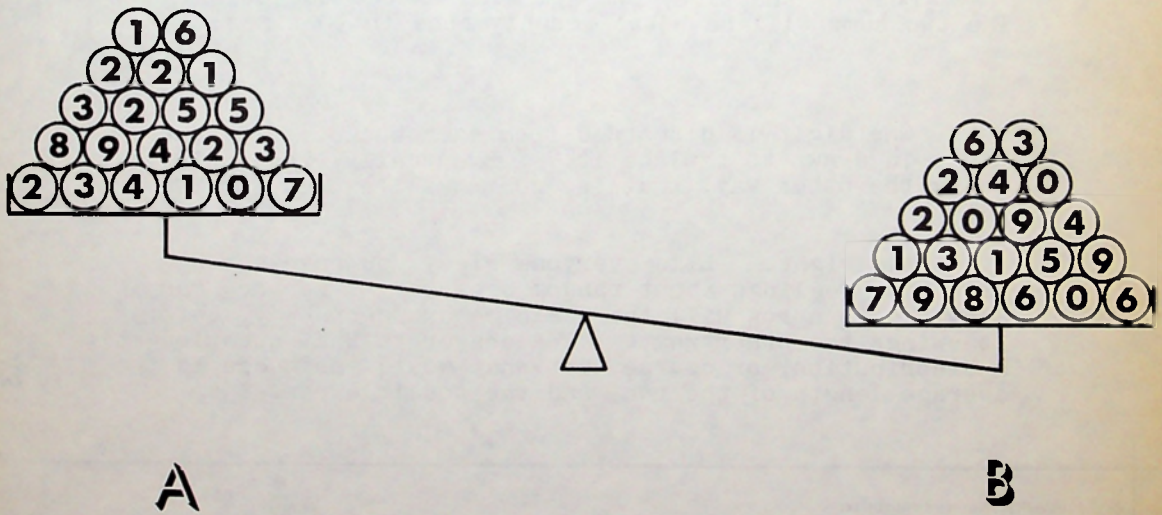
$2.50

67

October 1978 Volume 6 Number 10

Popular Computing

The world's only magazine devoted to the art of computing.



So you think you understand how random processes will behave?

Exploring Random Behavior--6

Let's consider a situation involving random digits. We can assume that we have available a subroutine that will produce for us, on demand, a single decimal digit and we further assume that these digits, in bulk, will pass any given test of randomness. The output of this subroutine should be like this sample of 450 successive calls:

```
919102907396070855447267238834079696068619658863985358979323846264338327950288419716939937510582097406250224971489682704605157172109191845329107693648896344446421948766181018598043688956549317247495448158753412727251735299123246675771853775370427813136975177338280384390284714101327120691215487836412467989197775775441621469891625491192562165228584964660423851305332281664761290981740144387702320818913060655544663191761741885138499175413232723385537
```

We now play a game that begins by drawing 20 digits for bucket A and 20 other digits for bucket B, as shown in the Figure on the cover of this issue. The sum of the digits in B exceeds the sum of the digits in A.

Most of the time, one sum will exceed the other. The two sums will be equal roughly once in 90 times.

One digit is discarded from each bucket, and a new digit is drawn to replace it. Eventually, the scale will swing the other way; that is, sum A will exceed sum B.

All right. Like everyone else, you have strong intuitive feelings about random behavior. For how many drawings in a row will the scale be swung one way, before it swings the other way? The answer to that question is a distribution, of course, but what would you guess is the average length of the run, and the possible range?

Publisher: Audrey Gruenberger
Editor: Fred Gruenberger
Associate Editors: David Babcock
Irwin Greenwald
Contributing Editors: Richard Andree
William C. McGee
Thomas R. Parkin
Advertising Manager: Ken W. Sim
Art Director: John G. Scott
Business Manager: Ben Moore

POPULAR COMPUTING is published monthly at Box 272, Calabasas, California 91302. Subscription rate in the United States is $20.50 per year, or $17.50 if remittance accompanies the order. For Canada and Mexico, add $1.50 per year. For all other countries, add $3.50 per year. Back issues $2.50 each.

After each swing of the scale, there are usually many short strings during which the scale swings back and forth, followed by a very long string. An experimental run, totalling 100 swings of the scale, showed an average length of 10.02 with a standard deviation of 12.3.

Is the number 20 a significant parameter of this problem? That is, would the results differ if the buckets held 15, or 25 numbers?

Another experimental run was made with the bucket size set at three. This showed the string length to be an average of 16.8 with a standard deviation of 15.1.

A reference was made above to tests of randomness. One such test is the Coupon-Collector's test, described in the January, 1955 issue of Mathematical Tables and Other Aids to Computation.

The idea of the test stems from an old probability problem: if numbered coupons are packed with a product, how many items of that product must one buy to obtain a complete set of the coupons? Children are introduced to this problem via baseball cards and other items that are packed in breakfast food boxes and on bubble gum wrappers.

For our random digits, we want to collect a complete set of ten digits. In the sample of 450 digits presented earlier, we find complete sets of digits as follows:

Set	1 in 19 successive digits,
2	25
3	36
4	20
5	39
6	36
7	19
8	49
9	21
10	27
11	63
12	23
13	30

with 43 digits left over that do not contain a complete set (the digit zero is missing).

The minimum possible size for a complete set is 10. The probabilities for each possible size of set have been calculated (to 32 significant digits!); a condensed table is given here:

10	.0004	20	.0415	30	.0343	40	.0150
11	.0016	21	.0436	31	.0320	41	.0136
12	.0042	22	.0447	32	.0298	42	.0124
13	.0080	23	.0451	33	.0276	43	.0112
14	.0130	24	.0447	34	.0255	44	.0102
15	.0186	25	.0438	35	.0234	45	.0092
16	.0244	26	.0424	36	.0215	46	.0083
17	.0298	27	.0406	37	.0197	47	.0075
18	.0346	28	.0387	38	.0180	48	.0068
19	.0385	29	.0365	39	.0164	49	.0062
						50	.0056

The principle involved in determining a "complete set" is common to many computing problems. The basic logic for testing for a complete set is shown in the accompanying flowchart, together with a scheme for tabulating the resulting distribution.

This logic was applied to a computer run of 1600 complete sets of random digits. The digits were generated from the following logic:

new d = fraction part of $(\text{old d} + p)^5$
where p = 3.14159265

new z = fraction part of $(\text{old z} + e)^5$
where e = 2.71828183

b = d + z

S = 10*fraction part(integer part(10000*b)/10).

(Initialize d and z to any value between zero and one.)

and the output from each iteration of all the above is one random digit, S.

The empirical results were:

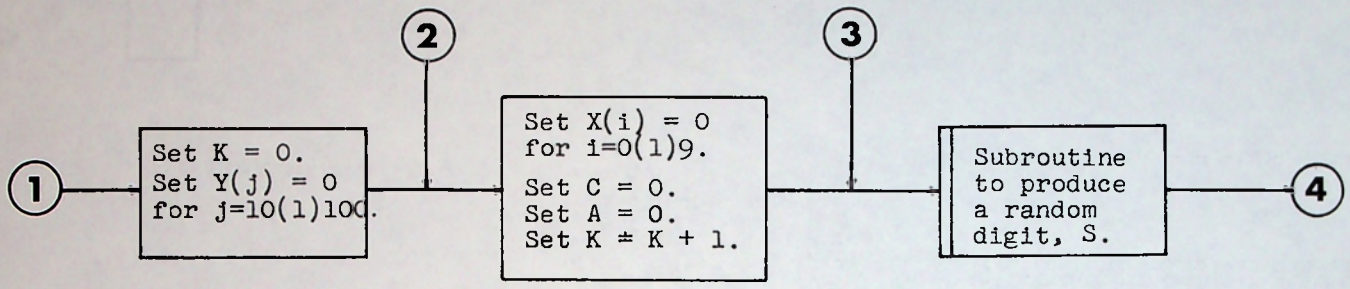
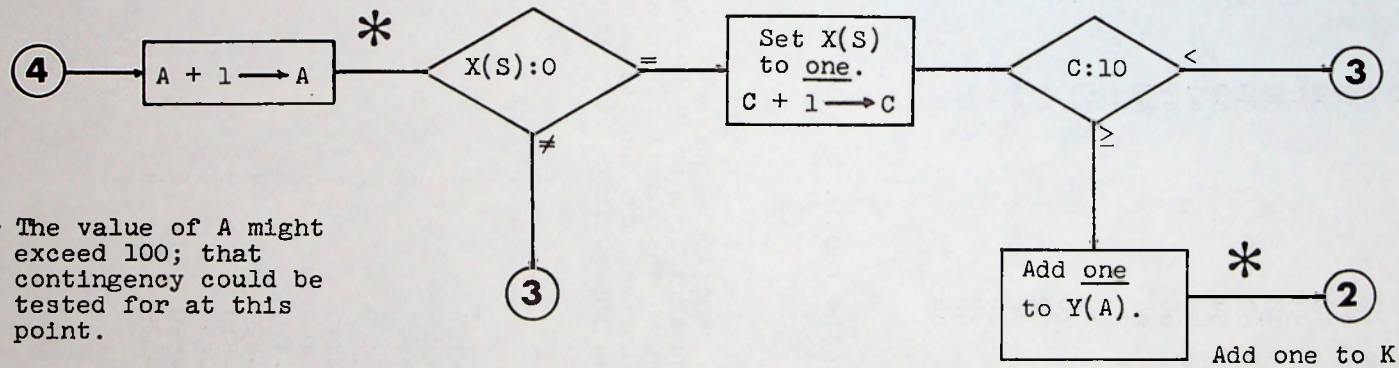


Table X holds one set of counters for 10 digits.
Table Y holds the distribution of results.

K counts the number of complete sets.
C counts to ten for one set.
A counts the length of the string.



The value of A might exceed 100; that contingency could be tested for at this point.

Add one to K here and test for the number of tests you wish to perform.

The Coupon-Collector's Test of Single Decimal Digits

10	0	31	49	52	2
11	1	32	46	53	9
12	8	33	39	54	11
13	12	34	27	55	3
14	21	35	53	56	4
15	42	36	30	57	6
16	38	37	31	58	7
17	38	38	39	59	1
18	62	39	26	60	3
19	54	40	18	61	1
20	69	41	21	62	1
21	68	42	26	63	1
22	64	43	17	64	2
23	80	44	12	65	0
24	87	45	15	66	1
25	76	46	11	67	1
26	62	47	14	68	0
27	75	48	16	69	2
28	53	49	10	70	3
29	52	50	5	71	0
30	44	51	15	72	1
				73	3

(with the remaining dozen strings longer than 73)

The same logic was applied to complete sets of the 2-digit numbers from 00 to 99, generated by a similar algorithm. The strings for complete sets were now quite long, of course, and the following distribution, divided into class intervals of 20, represents only 120 complete sets:

320-339	3	580-599	5
340-359	5	600-619	4
360-379	6	620-639	3
380-399	8	640-659	3
400-419	8	660-679	3
420-439	7	680-699	4
440-459	9	700-719	1
460-479	6	720-739	3
480-499	14	740-759	2
500-519	9	760-779	3
520-539	4	780-799	2
540-559	2	800-819	0
560-579	4	820-839	2
		(908)	1

homework 4

A Problem Of Interest

Assume that the U.S. Gross National Product (GNP) was 500 billion dollars in 1970, and was increasing at the steady rate of 3.57% per year. In 1970, the computing industry accounted for $6 billion of the GNP and was increasing at the steady rate of 11.84% per year. Now, assuming that these growth rates hold steady indefinitely, in what year:

(A) will the value of the computing industry equal or surpass the GNP?

(B) will the amount of growth for the year for the computing industry equal or surpass (for the first time) the amount of growth of the GNP?

Draw a flowchart for the logic of these two problems, and then write a Fortran program that follows that logic. Run the program and hand it in with the flowchart.

Attack these two problems in a step-by-step manner; that is, calculate all figures year by year for as many years as it takes. The object of the assignment is to see whether or not you have caught on to the rules of elementary Fortran (and to see, of course, if you can devise the proper logic for the solution to the problems). The Fortran programs developed in class, plus the list of Fortran rules, should give you everything you need. Check over your work for such things as:

Statements begin in column 7 (and you don't need printed coding sheets--plain paper will do nicely.

Fixed point variables (integers) must have names beginning with IJKLMN; floating point variables (reals) must have names beginning with the other letters of the alphabet. Be careful not to mix fixed and floating variables in the same statement--it indicates confused and muddled thinking.

(Please turn to page 15.)

A Rose Is A Rose...

There seemed to be much excitement generated in the United States at the news that a baseball player, Pete Rose, had completed a streak of 44 consecutive games in each of which he had had a safe hit.

The number of hits as a percentage of the number of times at bat is the player's Batting Average. A batting average of .333 is outstanding, meaning that for that player, he will score a safe hit once per every three times at bat, on the average. A batting average of .200 is very low. A safe hit gets the player on base without being walked (that is, on the basis of four balls) or without a sacrifice. A player will come up to bat perhaps as many as 5 times in one game, so that the probability of a hit at least once in a game can be calculated:

$$1 - (1 - BA)^5.$$

Thus, a player with a batting average of .333, coming to bat 5 times in one game, has a calculated probability of .868 of having at least one hit. This sort of reasoning leaves out all sorts of factors--particularly the psychological factors--and the conditions of the game. But, with this crude reasoning, that figure of .868 can be taken to mean "out of 100 games, the player will have a hit in 86 of them." Offhand, this would make it appear that a streak of 44 consecutive games with a hit is not very unusual.

Of course, the probability analysis just given is cold and unreal. It is like describing a series of coin tosses as coming out about 50% heads--there is then no information as to the strings of successive heads and tails that might have been registered during the run.

However, using only the information we have about batting averages, and weighting the whole thing in favor of getting a long run of games with hits, we can simulate the baseball situation with a computer run, to get a feel for the chances of such streaks. In what follows, the simulation is greatly simplified, and far removed from the real life situation. However, it will quickly demonstrate that a long run of hits is unlikely, based solely on the arithmetic of the batting average.

The scheme of the simulation is given in the accompanying flowcharts. Start with the flowchart for the One Game Subroutine. The Random Number Generator subroutine at Reference 4 will permit us some random variation. We postulate a generator whose output is uniformly distributed in the range from zero to 1.000. Thus, the decision at Reference 5 compares the number selected at random with the desired Batting Average (e.g., .300) and can be varied as you please. (22 out of 212 players in the major leagues have batting averages of .300 or better.) As shown in the flowchart, the player is allowed 5 times at bat in every game in this scheme, and we exit from the subroutine when a hit is scored or when the player has completed 5 times at bat.

The alert baseball fan will notice that our simulation is already wholly unreal, but at least is weighted toward having a winning streak.

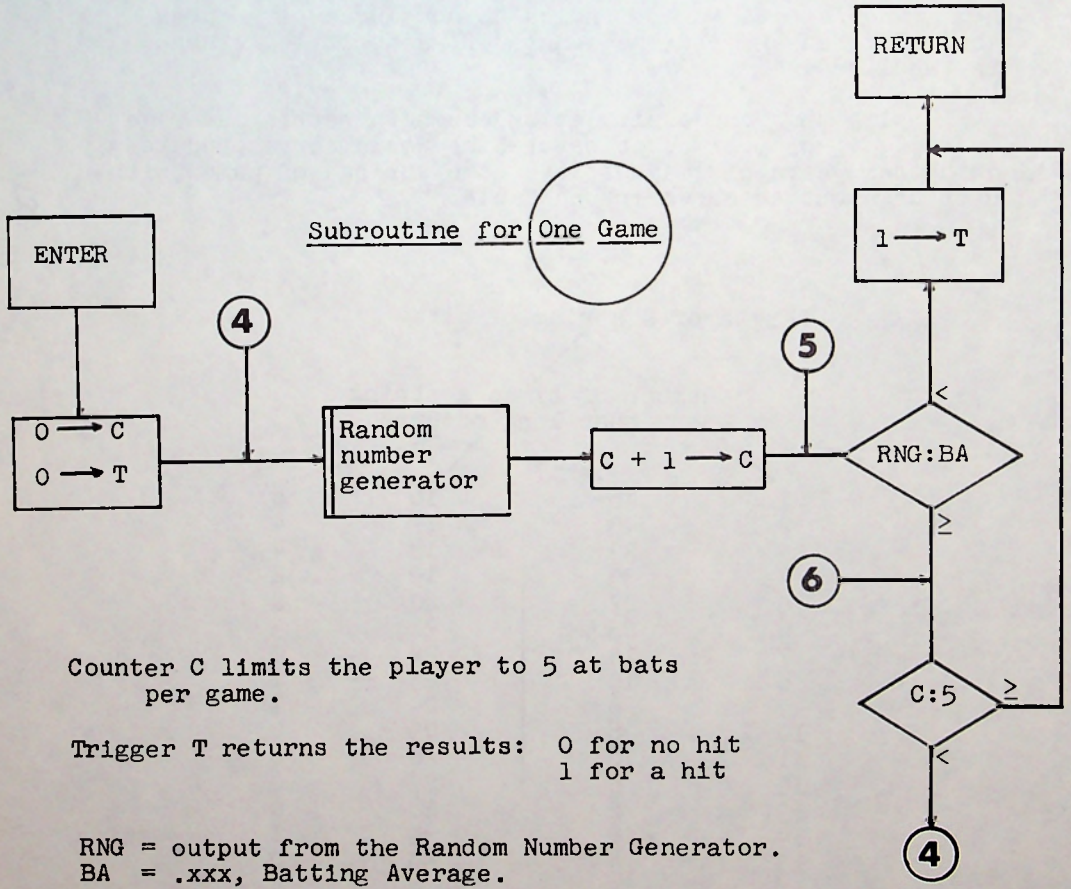
The Main flowchart indicates how successive games are triggered and how the length of each hitting streak is tabulated.

With this crude simulation, using a batting average of .300, in 6365 times at bat (which would correspond to about ten years of league play), the strings of games with hits came out as shown in the Table.

Length of a string of hits	Number of times a string that long occurred.		
1	46	16	6
2	45	17	4
3	39	18	4
4	33	19	2
5	27	20	2
6	23	21	3
7	14	22	0
8	19	23	1
9	6	24	1
10	6	25	0
11	10	26	0
12	6	27	0
13	2	28	0
14	8	29	1
15	4	30	0

All of this is unrealistic, for many reasons:

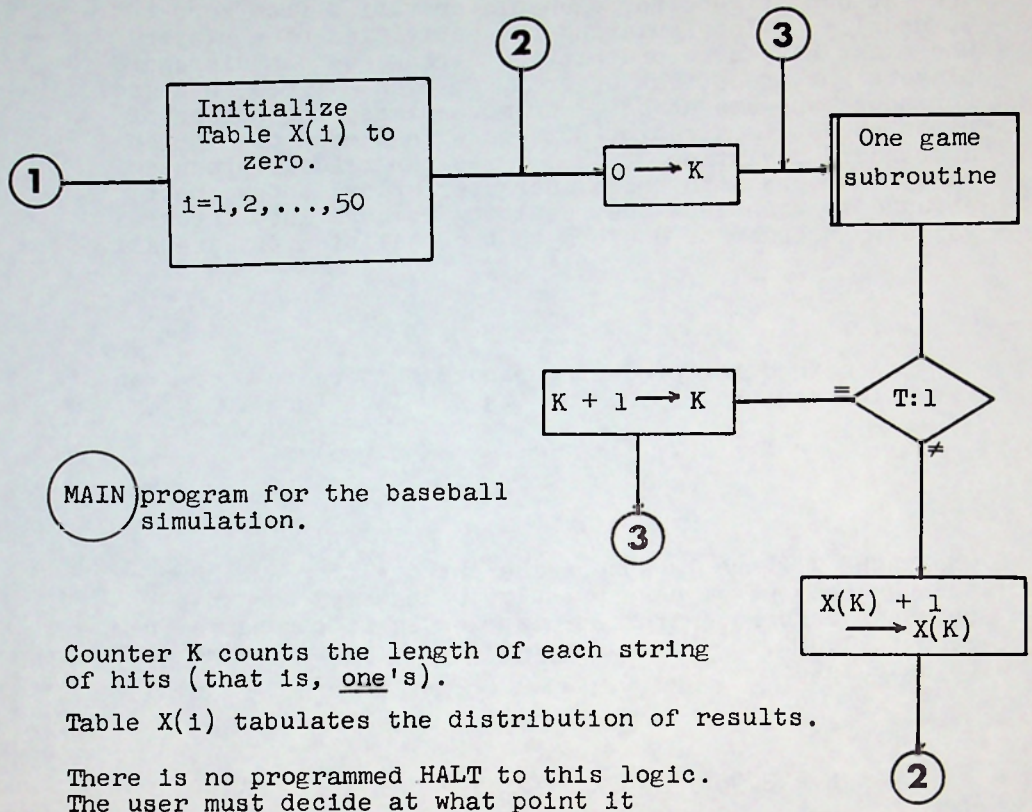
1. In real life, the record of successive games with hits must be established in one season. (The current record of 56 consecutive games with a hit was established by Joe DiMaggio in 1941.) A player could have 30 consecutive games at the end of one season, and 30 more at the start of the following season, and would probably not establish a record. Each season there are 162 games for each team, and a better computer simulation should take that into account.



Counter C limits the player to 5 at bats per game.

Trigger T returns the results: 0 for no hit
1 for a hit

RNG = output from the Random Number Generator.
BA = .xxx, Batting Average.



MAIN program for the baseball simulation.

Counter K counts the length of each string of hits (that is, one's).

Table X(i) tabulates the distribution of results.

There is no programmed HALT to this logic. The user must decide at what point it should be gracefully terminated (perhaps after 162 games).

2. The One Game flowchart allowed 5 times at bat in each game, which is totally unrealistic. This number is not a constant; it could in theory get as high as 7 in one game; it certainly varies from game to game, depending on many other conditions. The part of the flowchart at Reference 6 should be changed to use random numbers again to select a reasonable limit for each game played, with normal variation accounted for. This may be the stickiest part of a good simulation for this problem.

I have not been able to find out--from the baseball experts who abound--how one could calculate a mean and standard deviation for the number of times at bat in a game. Indeed, there seem to be differences of opinion as to what constitutes "at bat" in such a simulation.

If we examine the statistics printed in the papers, the "at bat" figure for each player varies from zero to 5, usually. In simulating the activities of a player who might have a record hitting streak, we can disregard players who are at bat 0, or 1, or even 2 times in a game. For those who are at bat 3 or more times, the number of times at bat is a set of numbers with a mean of 3.95 and a standard deviation of .85. We can readily produce random numbers with these characteristics, but we must arrange to translate the resulting numbers into integral values, in order to conform to the realities of baseball.

A method for producing random numbers to order was given in issue No. 55. In its simplest form it is:

$$R = M + S(X_1 + X_2 + \cdots + X_{12} - 6)$$

where the X's are 12 successive outputs from a generator such as the one we have postulated; namely, one that produces numbers uniformly distributed in the range from zero to 1.000. For our <u>times at bat</u> variable, we want to calculate:

$$R = 3.95 + .85(X_1 + X_2 + \cdots + X_{12} - 6)$$

The new variable is continuous, of course, but in any one baseball game it can only be an integer. The solution to that problem is to use $\lceil R \rceil$; that is, the greatest integer in R.

3. To be even more realistic, the batting average itself is not a constant, and particularly so during a hitting streak. In all likelihood, it would increase <u>during</u> the streak, and the simulation should reflect this fact.

It can be lots of fun to simulate this particular aspect of baseball, varying the parameters as you choose, and trying to get the simulation to match the real life situation as closely as possible. No matter how you stack the deck in favor of a run, it will turn out that a run of 44 games with safe hits is a rare item, and a run of 56 games is virtually impossible.

The VNR Concise Encyclopedia of Mathematics

Van Nostrand Reinhold Company

This book is a BEST BUY by any standard. The smallest of its virtues is its price: $14.95 for 816 pages (which is exceeded in value only by the NBS Handbook of Mathematical Functions at $12 for 1046 pages), nearly every one of which has an illustration and something highlighted in color.

The book started out as Kleine Enzyklopadie der Mathematik, followed by an English version under the title Mathematics at a Glance. The present edition (printed in the German Democratic Republic) dates from 1977.

This is a reference book of vital interest and utility to anyone involved with numerical work. In addition to providing the background, formulas, and derivations for any topic you can name, the entire world of mathematics, up to the level of a master's degree, is presented to some extent. The book invites browsing; open it at random and find out something you didn't know. Did you know, for example, that the basket-weaving method of evaluating 3 x 3 determinants is called Sarrus' Rule?

56 pages of photographs and drawings at the back of the book provide a graphic history of mathematics.

The German origin of the book comes through in occasional ponderous sentences, and in the list of contributors that includes:

Prof. H. Thiele — Dr. G. Wussing
Dr. H. Thiele — Prof. H. Wussing

There are flaws. For example, the account (on page 493) of the calculation by Shanks and Wrench of pi to 100265 decimal places is confused; the status of work on Fermat's conjecture is quite out of date. The flaws are extremely minor.

The bulk of the book is highly condensed but readily readable by anyone who has had high school mathematics and (for the advanced topics) a year of calculus. Great care has been exercised in selecting and drawing the hundreds of illustrations, and the added color on almost every page helps to clarify complicated topics.

A set of tables of elementary functions is included. These are rather poor (having 4 or 5 significant digits) and have no explanation of how to use them.

Still, this is a marvelous reference book--clearly one of those "should be on everyone's shelf" books. The difference is that this one won't stay on the shelf.

homework 5

You are to write and run an assembly language program that will read some data cards (at least 10 of them) each of which bears a 2-digit number. Each data item is to be printed as soon as it is read in. We want a count of the number of cards that are read, the sum of the data numbers, and the sum of squares of the data numbers.

The useful data consists of numbers between 10 and 99. We could test for end-of-file by detecting when there are no more cards to be read (the assembler's READ subroutine allows for this). A more graceful way (and one that will allow us to run several decks in one machine pass) is to have a sentinel card at the end of the data deck; this card can be punched with some large number, like 9999. Then the end-of-file condition can be tested for by comparing the input number to, say, 1000, and noting the end of the data deck when the condition is greater.

With the input data restricted to numbers under 100, calculate (by hand) the maximum number of data cards that can be in one deck before the sum of squares would cause an overflow.

Select data numbers for which you can easily predict the sum and sum of squares; you can thus test your program's logic as you debug it. Show how you tested your program.

(A) Draw a flowchart of the required logic.

Due date:_____

(B) Write the program and run it.

Due date:_____

SUMS and SUMS OF SQUARES

(Continued from page 7.)

There are two separate results required; be sure they are labelled (by writing on the printout, if necessary). It is possible to do the two problems simultaneously, but you will probably get there faster by treating them as two separate problems that can be done in one Fortran pass. Be sure to annotate your code with lots of COMMENTS cards.

The problems use the numbers 500 billion and 6 billion. Such numbers are readily handled in floating point notation; do not try to scale them down.

The result for part (A) above will represent sheer nonsense, of course, but notice that the nonsense is built into the problem.

1	1	2	3	5	8	13	21	34
1	2	3	5	8	13	21	34	55
1	3	(4)	7	11	18	29	47	76
1	4	5	(9)	14	23	37	60	97
1	5	6	11	(17)	28	45	73	118
1	6	7	13	20	(33)	53	86	139
1	7	8	15	23	38	(61)	99	160

In issue No. 54 we presented Les Marvin's problem, illustrated above, in which the circled terms from Fibonacci-type sequences form a new sequence, which is to be extended. A flowchart was given for a possible solution. In issue No. 57 a much simpler solution was given, due to John D. Beeby of Millbrae, California. We neglected to present Mr. Beeby's results:

4	1961	418299	73230466
9	3406	705479	122013961
17	5888	1187857	203125623
33	10137	1997018	337891627
61	17389	3352636	561650489
112	29733	5621097	932927398
202	50693	9412937	1548596408
361	86204	15744681	2568927609
639	146246	26307469	4258946341
1123	247577	43912648	7056700077

Printer's Glitch

Our issue No. 66 was delayed a week due to a rather interesting blunder on the part of the printer.

Each of the last 25 issues of POPULAR COMPUTING has appeared on five 11 x 17 sheets, making twenty 8 1/2 x 11 pages. We put the 8 1/2 x 11 masters together by twos, according to a fixed pattern; namely, the page numbers of the two sheets must add to 21, and the odd numbered sheet goes on the right.

The printer puts these double sheets together again by twos, in order to print on 17 x 22 sheets "two up," as they say. There are many ways in which this can be done. The layout shown at S on the facing page shows one way. The circled numbers represent the page numbers for the top side of the sheet; the bold numbers are then the pages on the back side.

The layout shown at T represents how issue No. 66 first arrived. Minor lapses are not unknown in the printing business--say, one sheet not backed up correctly--but this printing was unusual: <u>every</u> page was wrong.

The platemaker cannot get the adjacent pages out of order; that is, he cannot separate page 3 from page 18, since they are securely fastened to each other. He can, however, mix up everything else.

PROBLEM 246

Problem: How many ways are there to put the sheets together correctly in order to print two up? In how many ways can it be done wrong? ...and how many of <u>those</u> have every page backed up incorrectly?

There is also one minor problem here: Is there some simple rule for putting the sheets together, corresponding to the rule given above for tying two pages together?

PC67-17

S
T
19
20
4
3
2
1
17
18
15
16
8
7
6
5
13
14
11
12
10
9
12
11
9
10
6
15
12
9
9
6
15
12
17
20
8
3
4
1
13
18
19
16
10
7
2
5
11
14

ROOTS

$$\sqrt[2]{2} + \sqrt[3]{2} \geq 2$$

$$\sqrt[2]{3} + \sqrt[3]{3} \geq 3$$

$$\sqrt[2]{4} + \sqrt[3]{4} + \sqrt[4]{4} \geq 4$$

$$\sqrt[2]{5} + \sqrt[3]{5} + \sqrt[4]{5} \geq 5$$

$$\sqrt[2]{6} + \sqrt[3]{6} + \sqrt[4]{6} + \sqrt[5]{6} \geq 6$$

* * * * * * * * * * * * * * *

$$\sqrt[2]{9} + \sqrt[3]{9} + \sqrt[4]{9} + \sqrt[5]{9} + \sqrt[6]{9} \geq 9$$

The table above shows what happens when we sum the successive roots of an integer, N. It takes <u>two</u> roots of 2 to sum to at least 2; <u>three</u> roots of 4 to sum to at least 4; <u>five</u> roots of 9 to sum to at least 9; and so on.

We can build a table:

To sum to at least ⇩	Takes this many successive roots ⇩	
3	2	1.500
4	3	1.333
9	5	1.800
13	7	1.714
30	17	1.764
100	70	1.429
200	157	1.274
500	435	1.149

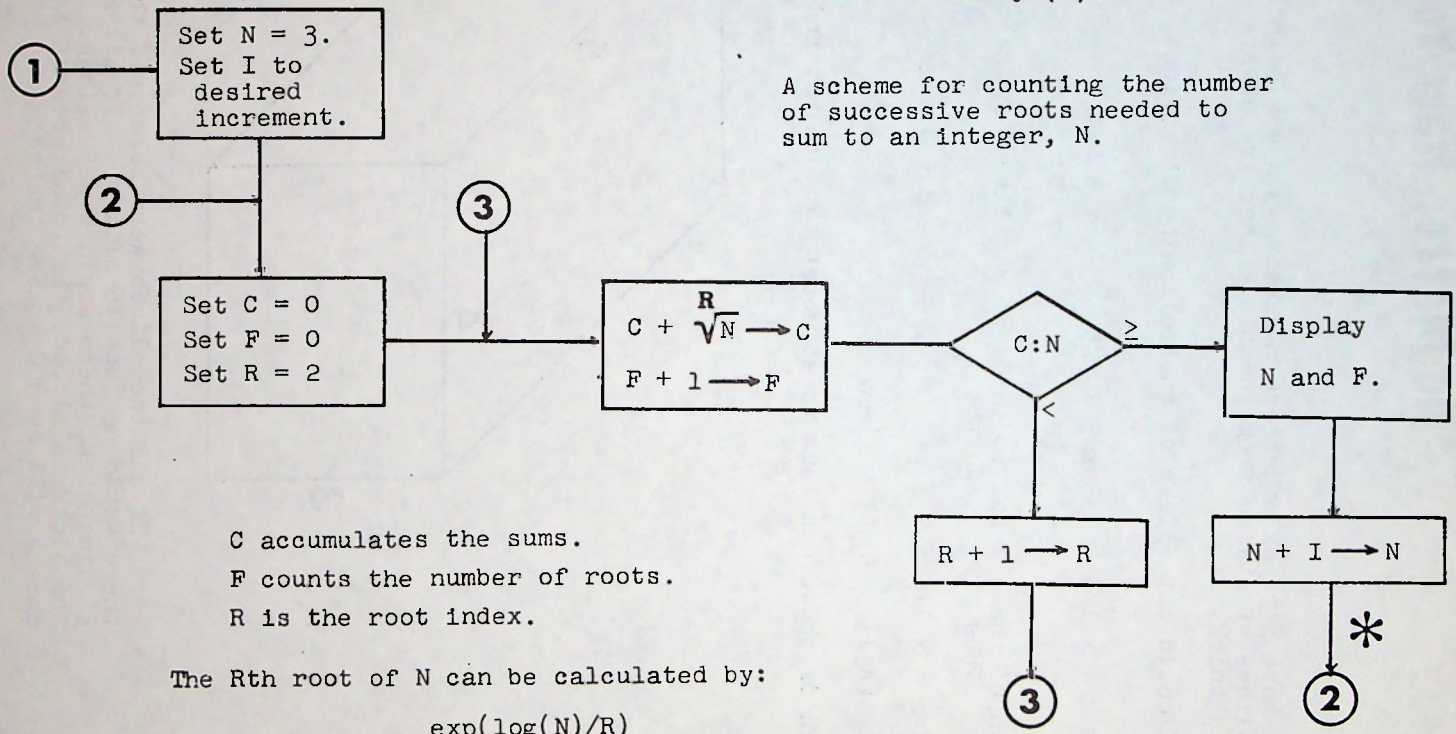
The third column is the ratio of the first two columns. After some minor fluctuations due to small integers, this ratio settles down and gets continuously smaller. The accompanying flowchart indicates a way to extend the table indefinitely.

As the number N increases, it takes more and more of its roots to sum to N. How large must N get so that it takes <u>N</u> roots to sum to N?

That's foolish--it can never happen. What <u>can</u> happen, though, is that that ratio can get as close to 1.000 as you please. For what value of N will it first get to 1.1000?

No stopping point has been indicated on this flowchart; a test for the number of values of N can be inserted at the point marked by (*).

A scheme for counting the number of successive roots needed to sum to an integer, N.



C accumulates the sums.

F counts the number of roots.

R is the root index.

The Rth root of N can be calculated by:

$$\exp(\log(N)/R)$$

No Matter How You Slice It

PROBLEM 248

I have a block of cheese measuring 3 x 4 x 5 inches. I cut slices of cheese from the block that are always 1/5 inch thick.

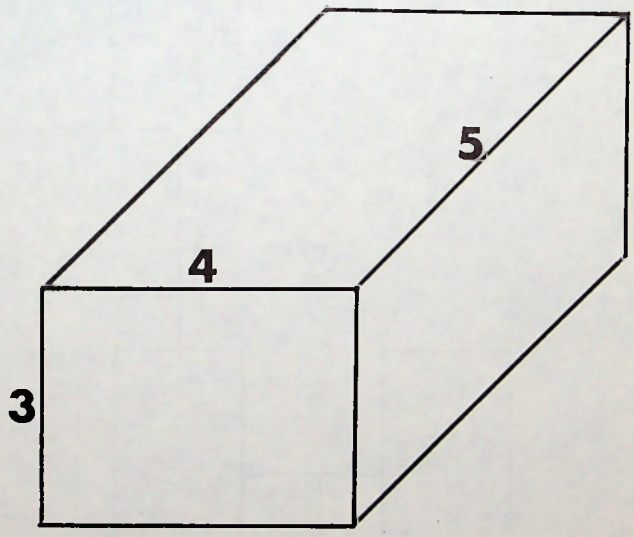
I could cut slices of the following dimensions, to start:

3 x 4 x 1/5	3 x 5 x 1/5	4 x 5 x 1/5
(A)	(B)	(C)

but each of these leads to three possibities for the next slice. Thus, (A) leads to:

3 x 4 x 1/5	3 x 4.8 x 1/5	2.8 x 4 x 1/5
(A_1)	(A_2)	(A_3)

and each of these can lead to at least two others, and so on.



If all the slices were taken off the 3" dimension, there would be just 15 slices. If all were taken off the 5" dimension, there would be 25 slices. No matter how the block is sliced, the total area of 1/5" pieces of cheese will be 300 square inches.

Problem: How many <u>different</u> pieces of cheese could be cut? (Probably several million.)